How to work with HotSpot

Deprecated

This page is deprecated and it's content is either old or being moved to the OpenJDK Developers' Guide. Please update your links.

Quick summary

- **Post hs integration**
  - When a change is pushed to the jdk/hs forest, an automatic test cycle is triggered. Developers who have pushed new changes must be available to deal with any failures in the following test cycle. This is basically the same tests that is run in the submit forest. There is also a larger set of tests that is run every night, meaning that it is also good practice to be available to respond to any failures in the following nightly test run.
- **New bugs / Triage**
  - The gatekeeper will monitor the nightlies and files bugs in JBS for any issues not already handled by developers.
  - Initial bug triage is handled by each component team. It is recommended to include both engineers with experience of our test infrastructure and senior JVM developers in the triage meetings to allow for fast and accurate analysis of new failures. It is also recommended that the triage team has the mandate to assign issues to other members of the component team.
- **Each failure falls into one of the following categories**
  - **Product bug or Test bug.** Such an issue can be resolved with either a new change that solves the issue or by backing out the change that caused the regression. This category includes both regressions and new tests and features which cause tests to fail. One should **not** quarantine a test as a solution to a new product issue. It is expected that a quick fix or applying the anti-delta will be the common approach.
  - **Infrastructure issue.** Make sure bugs are filed, both for job failures and for test failures. Then wait until a new test run is executed correctly. If we have severe infra issues it might be recommended to not allow any new changes until the infra issues are resolved.
  - **Intermittent issue.** Sometimes we see intermittent failures due to old issues. If the only timely approach is to open a new bug and to quarantine the test, then it should be done. But quarantining tests comes with a risk due to reduced test coverage.

Pushing changes?

- Pushing a HotSpot change
- Submit Repo

Filing bugs?

- Filing bugs for HotSpot failures
- HotSpot JBS components
- Identifying and handling a critical failure

Triaging bugs?

- Bug triage
- Integration and Integration Blockers

Looking for answers?

- HotSpot How To

Hotspot Gatekeeping Policy - Three Golden Rules

A critical failure must be handled immediately

Analyzing and handling critical failures has high priority. The expected turnaround time to resolve an issue is before the next nightly starts, but should be handled in the most reasonable fashion, given our distributed community. Handling a failure means fixing the bug that caused the failure or backing out the change that caused the failure. In exceptional cases it can mean quarantining the test or ask for quarantine exception, especially reasonable in case of long time intermittent issues.

Pushes into a repository with existing critical failures must be done with care

How strict to be is decided on a case by case basis, but all pushes must be done with care, since failing tests may be harder to isolate and handle when more changes are coming in. If the situation is too bad the repository can be locked, but if the issue is isolated and the risk for complications is low, it may be better not to block additional changes being pushed.

No integration from a repository with new failures

A repository where new failures (a.k.a. integration blockers) have been seen in testing must not be pushed upstream. Failures must not spread and interfere with other engineers' development. This goes for both project repositories that integrate with `jdk/hs`, and `jdk/hs` itself as it integrates with `jdk/jdk`. Please note that there is no difference between product bugs and test bugs when it comes to integration blockers.